prudsys AG
P60074US

Verfahren und Vorrichtung zum Ermitteln einer Menge großer Sequenzen aus einer elektronischen Datenbank

Die Erfindung liegt auf dem Gebiet der Sequenzanalyse in elektronischen Datenbanken.

Hintergrund

Sequenzanalysen extrahieren Sequenzen von Elementen aus transaktionalen elektronischen Daten. In dem Maße, in dem transaktionale Daten zunehmend elektronisch erfasst werden, gewinnen Sequenzanalysen an Bedeutung.

Ein Anwendungsbeispiel für Sequenzanalysen stellt die Clickstreamanalyse dar, in welcher typische Nutzerpfade von Nutzern von Websites berechnet werden. Weitere Beispiele von Sequenzanalysen sind die Textanalyse zur Extraktion charakteristischer Wortfolgen in elektronisch gespeicherten Dokumenten sowie die Warenkorbanalyse unter Berücksichtigung der Produktreihenfolge im Handel zur Bestimmung typischer Produktkettenkäufe. Vielfältige Anwendungen der Sequenzanalyse finden sich ferner in der Chemie sowie in der Genetik.

Derzeit existieren eine Vielzahl spezialisierter Verfahren zur Sequenzanalyse, insbesondere im Bereich der Genetik, jedoch sind nur wenige universelle verwendbar. Im einfachsten Fall werden hierbei alle Varianten möglicher Sequenzen auf deren Häufigkeit untersucht, doch ist dies aus Gründen der Rechenkapazität nur für kleine Datenmengen möglich. Eine Alternative sind Sequenzanalysealgorithmen, welche auf Suchbäumen basieren, beispielsweise der Algorithmus „Capri" der Firma Lumio (http://www.spss.com/PDFs/CP2SPCZ-1201.pdf). Doch ist auch deren Geschwindigkeit zur Analyse großer Transaktionsdaten unzureichend. Eine schnellere Alternative stellen Verfahren dar, welche auf Ideen der klassischen Warenkorbanalyse beruhen und bereits für die sequentielle Warenkorbanalyse genutzt wurden (Agrawal R., Srikant R. Mining Sequential Patterns. IBM Almaden Research Center; 650 Harry Road, San Jose).

Die Erfindung

Aufgabe der Erfindung ist es, ein Verfahren und eine Vorrichtung zum Ermitteln einer Menge großer Sequenzen aus einer elektronischen Datenbank anzugeben, mit denen die Nachteile

des Standes der Technik überwunden werden, insbesondere eine effiziente Sequenzanalyse bei großer Datenmengen in verschiedenen Anwendungsfällen ausgeführt werden kann.

Nach einem Aspekt der Erfindung ist ein Verfahren zum Ermitteln einer Menge großer Sequenzen aus einer elektronischen Datenbank mit einer Folge D={d1, ..., dn} von Transaktionen di ($1 \leq i \leq n$) in einem Computersystem mit einem implementierten Abfragemodul geschaffen, wobei jede der großen Sequenzen auf der Folge D von Transaktionen di einen Stützwert aufweist, der größer oder gleich einem vorgegebenen Stützwert S ist, wobei jede der Transaktionen di der Folge D eine Sequenz von Elementen eines Satzes E={e1, ..., em} von Elementen ej ($1 \leq j \leq m$) ist und wobei das Verfahren die folgenden Schritte umfaßt:

a) Ermitteln einer Menge L1 großer Sequenzen aus der Folge D von Transaktionen, wobei die großen Sequenzen der Menge L1 jeweils genau ein Element des Satzes E umfassen und ein zugehöriger Stützwert S_{L1} auf der Folge D von Transaktionen jeweils größer oder gleich dem vorgegebenen Stützwert S ist;

b) Ermitteln einer Menge L2 großer Sequenzen aus der Folge D von Transaktionen, wobei die großen Sequenzen der Menge L2 jeweils genau zwei Elemente des Satzes E in einer jeweiligen Reihenfolge R_{L2} umfassen und ein zugehöriger Stützwert S_{L2} auf der Folge D von Transaktionen jeweils größer oder gleich dem vorgegebenen Stützwert S ist und wobei beim Ermitteln der Menge L2 großer Sequenzen nur Sequenzen berücksichtigt werden, die als Teilsequenz eine der großen Sequenzen der Menge L1 umfassen;

c) Ermitteln einer Menge Lk (k > 2) großer Sequenzen aus der Folge D von Transaktionen, wobei die großen Sequenzen der Menge Lk jeweils genau k Elemente des Satzes E in einer jeweiligen Reihenfolge R_{Lk} umfassen und ein zugehöriger Stützwert S_{Lk} auf der Folge D von Transaktionen jeweils größer oder gleich dem vorgegebenen Stützwert S ist, wobei beim Ermitteln der Menge Lk nur Sequenzen berücksichtigt werden, die als sich teilweise überlappende Teilsequenzen zwei der großen Sequenzen der Menge Lk-1 mit der jeweiligen Reihenfolge RLk-1umfassen; und

d) Wiederholen des Schrittes c) für k=k+1 und Abbrechen des Wiederholens des Schrittes c) beim Erfüllen einer vorgegebenen Abbruchbedingung.

Nach einem anderen Aspekt der Erfindung ist eine integriertes Sequenzanalysesystem geschaffen, mit:

- einer elektronischen Datenbank mit einer Folge D={d1, ..., dn} von Transaktionen di ($1 \leq i \leq n$), wobei jede der großen Sequenzen auf der Folge D von Transaktionen di einen Stützwert aufweist, der größer oder gleich einem vorgegebenen Stützwert S ist, wobei jede der Transaktionen di der Folge D eine Sequenz von Elementen eines Satzes E={e1, ..., em} von Elementen ej ($1 \leq j \leq m$) ist;
- einem Abfragemodul, welches eine mit der Datenbank gekoppelte Abfrageeinrichtung und eine Verarbeitungseinrichtung zum Erfassen von Abfrageparametern und zum Erzeugen von Abfragen an die Abfrageeinrichtung umfaßt;
- Mitteln zum Ermitteln einer Menge L1 großer Sequenzen aus der Folge D von Transaktionen, wobei die großen Sequenzen der Menge L1 jeweils genau ein Element des Satzes E umfassen und ein zugehöriger Stützwert S_{L1} auf der Folge D von Transaktionen jeweils größer oder gleich dem vorgegebenen Stützwert S ist;
- Mitteln zum Ermitteln einer Menge L2 großer Sequenzen aus der Folge D von Transaktionen, wobei die großen Sequenzen der Menge L2 jeweils genau zwei Elemente des Satzes E in einer jeweiligen Reihenfolge R_{L2} umfassen und ein zugehöriger Stützwert S_{L2} auf der Folge D von Transaktionen jeweils größer oder gleich dem vorgegebenen Stützwert S ist und wobei beim Ermitteln der Menge L2 großer Sequenzen nur Sequenzen berücksichtigt werden, die als Teilsequenz eine der großen Sequenzen der Menge L1 umfassen;
- Mitteln zum Ermitteln einer Menge Lk ($k > 2$) großer Sequenzen aus der Folge D von Transaktionen, wobei die großen Sequenzen der Menge Lk jeweils genau k Elemente des Satzes E in einer jeweiligen Reihenfolge R_{Lk} umfassen und ein zugehöriger Stützwert S_{Lk} auf der Folge D von Transaktionen jeweils größer oder gleich dem vorgegebenen Stützwert S ist, wobei beim Ermitteln der Menge Lk nur Sequenzen berücksichtigt werden, die als sich teilweise überlappende Teilsequenzen zwei der großen Sequenzen der Menge Lk-1 mit der jeweiligen Reihenfolge R_{Lk-1} umfassen; und
- Mitteln zum Wiederholen des Schrittes c) für k=k+1 und Abbrechen des Wiederholens des Schrittes c) beim Erfüllen einer vorgegebenen Abbruchbedingung.

Im Unterschied zur klassischen Warenkorbanalyse wird bei der Erfindung die Reihenfolge der Elemente in den Transaktionen berücksichtigt. Mit Hilfe der Erfindung können umfangreiche elektronische Datenmengen nach charakteristischen Sequenzen effizient hinsichtlich des benötigten Zeitaufwands und der notwendigen Rechenkapazität automatisch ausgewertet

werden. Hierdurch wird erstmalig die sequentielle Analyse großer Datenmengen möglich, wie sie beispielsweise Logfiles stark frequentierter Webserver aufweisen.

Zeichnung

Die Erfindung wird im folgenden anhand von Ausführungsbeispielen unter Bezugnahme auf eine Zeichnung näher erläutert. Hierbei zeigen:

Figur 1 eine schematische Darstellung zur Erläuterung eines Verfahrens zur Sequenzanalyse nach der Erfindung in Verbindung mit einer Clickstreamanalyse;

Figur 2 eine schematische Darstellung einer Anordnung zum Ausführen eines Verfahrens zur Sequenzanalyse nach der Erfindung;

Figur 3 ein Ablaufdiagramm für eine Ausführungsform eines Verfahrens zur Sequenzanalyse nach der Erfindung gemäß einer ersten Variante; und

Figur 4 ein Ablaufdiagramm für eine Ausführungsform eines Verfahrens zur Sequenzanalyse nach der Erfindung gemäß einer zweiten Variante.

Beispielhafte Ausführungsformen

Im folgenden werden ein Verfahren und eine Vorrichtung zum Ermitteln einer Menge großer Sequenzen aus einer elektronischen Datenbank unter Bezugnahme auf die Figuren 1 bis 4 beschreiben.

Zunächst soll das bei einer Sequenzanalyse zu lösende Problem im Detail erläutert werden. Es sei E = {e1, e2, ..., em} ein Satz von Literalen, welche als Elemente bezeichnet werden. Eine Sequenz $<s_0, s_1, \ldots, s_k>$ sei eine geordnete Liste von Elementen. Es sei D={d1, d2, ..., dn} ein Satz von Transaktionen, wobei jede Transaktion di ($1 \leq i \leq n$) eine Sequenz von Elementen aus E ist, so daß gilt: $di \subseteq E$. Jeder Transaktion di wird ein eindeutiger Identifikator zugeordnet, gekennzeichnet als „di-ID".

Die Sequenz <a0, a1, ..., ak> ist enthalten (Teilsequenz) in einer anderen Sequenz <b0, b1, ..., bl>, wenn natürliche nichtnegative Zahlen $i_1 < i_2 < \ldots < i_k$ existieren, so daß

$a1 = bi_1$; $a2 = bi_2$; ...; $ak = bi_k$

Es wird definiert, dass eine Sequenz A eine Stützung (support) s% auf der Menge der Transaktionen D besitzt, wenn sie in s% aller Transaktionen der Menge D enthalten ist. Wir bezeichnen eine Sequenz als groß, wenn gilt, dass ihr Support nicht kleiner als ein vorgegebener Minimalwert ist. Eine Sequenz der Länge k wird als k-Sequenz bezeichnet. Im weiteren wird die Menge aller k-Sequenzen als Lk bezeichnet. Die Menge aller k-Sequenzen, welche potenziell groß sein können, wird als Ck bezeichnet.

Die Menge E kann eindeutig auf eine Untermenge natürlicher Zahlen abgebildet werden, das heißt, jedem Element aus E kann ein eindeutiger natürlicher Identifikator zugeordnet werden.

Unter Benutzung der eingeführten Begriffe kann die Aufgabe folgendermaßen formuliert werden. Auf Basis der vorgegebenen Mengen E und D sollen alle Sequenzen gefunden werden, deren Supportwert größer oder gleich einem vorgegebenen Minimalwert ist, das heißt, es wird die folgende Menge gesucht:

$$L = \bigcup_k L_k$$

Ein bekannter Ansatz zur Lösung dieser Aufgabe besteht in der Auflistung aller möglichen Sequenzen aus Elementen der Menge E und dem Auszählen der sie enthaltenden Transaktionen. Es wird die Anzahl aller möglichen Sequenzen aus Elementen der Menge E mit der Länge von 1 bis k gefunden. Die Mächtigkeit der Menge E ist m. Die Anzahl der Sequenzen der Länge 1 ist gleich der Anzahl der Elemente der Menge E, also m. Die Anzahl aller möglichen Sequenzen der Länge i aus m Elementen ist m^i. Dann ist die Anzahl aller Sequenzen N gleich der Summe der geometrischen Reihe zur Basis m:

$$N = \sum_{i=1}^{k} m^i = \frac{m^{k+1} - m}{m - 1}.$$

Bereits für m = 100 und k = 10 erhalten wir $N = \frac{100^{11} - 100}{100 - 1} \approx 10^{20}$ verschiedene Sequenzen. Es ist klar, dass ein solcher Ansatz zur Lösung der Aufgabe unakzeptabel ist. Für reale Daten sind in allen Transaktionen der Menge D deutlich weniger Sequenzen enthalten.

Weit logischer und schneller ist der Test aller Sequenzen, welche in jeder Transaktion enthalten sind und die Bestimmung des Supportwertes für jede von ihnen. Angenommen, die Länge der Transaktion d0 ist gleich m. Dann enthält die Transaktion d0 die Anzahl aller Sequenzen der Länge i, welche gleich der Menge der i Kombinationen der Elemente über m ist. Dann ist die gesamte Anzahl der Sequenzen N, welche in einer Transaktion der Länge m enthalten sind, gleich:

$$N = \sum_{i=1}^{m} \frac{m!}{i!(m-i)!} = 2^m - 1$$

Eine Transaktion der Länge 10 enthält 1023 Sequenzen, eine Transaktion der Länge 20 enthält 1048575 Sequenzen der Längen von 1 bis 10 und entsprechend 20. Obwohl die angeführten Zahlen den ungünstigsten Fall beschreiben, kann die Anzahl der möglichen Sequenzen, welche in einer Transaktion enthalten sind, dennoch sehr groß werden.

Jede dieser Sequenzen muß auf ihre minimale Supportbedingung getestet, das heißt, es muß die Summe der sie enthaltenen Transaktionen gebildet werden. Es ist klar, dass dieser Prozeß ebenfalls sehr zeitintensiv ist. In den hier beschriebenen Verfahren nach der Erfindung wird deshalb ein Ansatz verwendet, welcher keine überflüssigen Tests benötigt.

Im folgenden werden zwei Variante der Extraktion großer Sequenzen beschrieben. Beide nutzen einen gemeinsamen Satz grundlegender Ideen, aber unterscheiden sich in der Summation der Supportwerte der Sequenzen.

Die Verfahren gehören zur Klasse der Iterationsverfahren. In jedem Arbeitsschritt der Verfahren werden alle großen Sequenzen einer gegebenen Länge bestimmt. Die Arbeit der Verfahren dauert solange, bis alle großen Sequenzen maximaler Länge gefunden wurden, das heißt, bis im vorliegenden Arbeitsschritt keine weiteren großen Sequenzen gefunden werden. Ein anderes Kriterium zur Terminierung des Verfahrens kann das Erreichen einer maximalen Länge der Sequenzen sein, welche vom Nutzer vorgegeben worden ist.

In beiden Varianten des Verfahrens werden die großen Sequenzen der Länge k benutzt, um die Menge neuer potentieller großer Sequenzen der Länge k+1 zu konstruieren. Wir bezeichnen eine Sequenz als Kandidat, wenn diese aus großen Sequenzen der um Eins verminderten

Länge generiert worden ist und möglicherweise ebenfalls eine große Sequenz darstellt. Auf diese Weise bildet die Menge der Kandidaten der Länge k die Menge Ck. Offensichtlich gilt: $Ck \supseteq Lk$.

1. Methode zur Generierung der Kandidatenmenge

Beide Verfahren nutzen das gleiche Prinzip der Generierung von Kandidaten, welches sich etwas von dem der Generierung der Kandidaten im oben genannten Ansatz von IBM unterscheidet. Die Funktion zur Generierung der Kandidaten nutzt als Eingangsmenge alle großen k-Sequenzen – Lk. Sie liefert im Ergebnis eine Supermenge der Mengen aller großen (k+1)-Sequenzen – Ck+1. Unter Nutzung einer SQL-artigen Syntax kann das Verfahren der Generierung der Kandidatenmenge folgendermaßen aufgeschrieben werden:

INSERT INTO Ck
SELECT $p.item_1, p.item_2, \ldots, p.item_k, q.item_k$
FROM p,q
WHERE $p.item_2 = q.item_1$, $p.item_3 = q.item_2$, …, $p.item_k = q.item_{k-1}$

Die Arbeitsweise der Generierung der Kandidaten kann anhand folgenden Beispiels veranschaulicht werden.

Angenommen, in einem Schritt der Arbeit des Verfahrens sind folgende große Sequenzen der Länge 4 gefunden worden:

1. 1 3 2 6
2. 1 4 1 2
3. 2 1 3 2
4. 2 2 2 2
5. 3 1 2 1
6. 3 2 6 2
7. 4 1 2 1

Mittels Durchgehen aller möglichen Paare großer 4-Sequenzen, beginnend mit dem ersten, und unter Einschluß auch aller aus der gleichen Sequenz bestehenden, erhält man folgende Kandidatenmenge:

1 3 2 6 2 – Ergebnis der Vereinigung von 1 und 6
1 4 1 2 1 – Ergebnis der Vereinigung von 2 und 7
2 1 3 2 6 – Ergebnis der Vereinigung von 3 und 1
2 2 2 2 2 – Ergebnis der Vereinigung von 4 und 4

Die Sequenzen 5., 6. und 7. lieferten kein einziges Paar, in welchem sie die ersten der „Eltern" wären; die Sequenz 5. war überhaupt nicht an der Generierung von Kandidaten beteiligt; die Sequenz 4. wurde mit sich selbst vereinigt.

Im Ergebnis der Arbeit des Verfahrens kann eine Situation entstehen, in der mittels dieser Methode keine einzige Kandidaten-Sequenz generiert wird. Das bedeutet, dass es nicht möglich ist, auch nur eine einzige Sequenz des nächsten, k+1-ten, Levels zu finden. Folglich terminiert das Verfahren in diesem Schritt.

Im dargelegten Beispiel wurden zur Konstruktion der Kandidatenmenge alle möglichen Paare großer Sequenzen des vorhergehenden Levels herangezogen. Während der Arbeit des Verfahrens auf realen Daten kann es leicht passieren, in einem gewissen Schritt eine Menge aller großen Sequenzen zu finden, welche aus Tausenden oder Zehntausenden Sequenzen besteht. Der Prozeß der vollständigen Durchsicht aller Paare besitzt eine quadratische Komplexität. Daher müsste man auf realen Daten Millionen oder Hunderte von Millionen Operationen zum Vergleich von Sequenzen ausführen. Das kann zu ernsthaftem Zeitaufwand für den Prozeß der Kandidatengenerierung führen.

Daher wurde eine Datenstruktur erarbeitet, welche es erlaubt, die Komplexität der Durchsicht der Paare großer Sequenzen zu verringern. Gemäß dieser Idee ist es notwendig, für jede große k-Sequenz die erste und die letzte große (k+1)-Sequenz zu speichern, welche mit der betrachteten k-Sequenz beginnen. Natürlich ist es dafür notwendig, daß der Satz der k-Sequenzen von den kleinen k-Sequenzen zu den großen Sequenzen sortiert wird. Die k-Sequenz A wird als kleiner als die k-Sequenz B definiert, wenn das erste verschiedene Element in der Sequenz A kleiner als in der Sequenz B ist.

Somit besitzen wir in einem Schritt des Verfahrens Lk – die Menge der großen k-Sequenzen des vorangegangenen Schrittes, Ck+1 – die Menge der Kandidaten. Weiterhin wird für jede k-Sequenz aus Lk die Nummer des zweiten „Elternteils" gespeichert, welcher diese Sequenz generiert hat. Unter Benutzung dieser Nummer kann für jede k-Sequenz gleich das Spektrum der Sequenzen aus Lk bestimmt werden, mit welchen diese vereinigt werden kann. Diese Datenstruktur wird in beiden Varianten des beschriebenen Verfahrens genutzt. Zur Speicherung der benötigten Nummern für die Sequenzen aus Lk-1 wird ein dynamisches Array der Dimension |Lk-1|×2 genutzt.

Im k-ten Schritt liegt die Menge der großen Sequenzen Lk vor, mit jeder von denen die Nummer derjenigen Sequenz aus Lk-1 verbunden ist, welche den zweiten „Elternteil" bildet. Im Ergebnis der Arbeit erhält man im k-ten Schritt aus Ck+1 die Menge Lk+1, für deren Sequenzen die Nummer ihres zweiten Elternteils gespeichert werden, und es wird eine neue Struktur aus den Nummern der Sequenzen aus Lk geformt.

Im oben angeführten Beispiel wurde die Sequenz <2 1 3 2> im vorhergehenden Schritt aus den Sequenzen <2 1 3> und <1 3 2> geformt. Angenommen, die Sequenz <1 3 2> stand in der Nummer 3 der Liste der Sequenzen L3. Dann müssen in der beschriebenen Struktur für die Sequenz der Nummer 3 die Nummern 1 und 1 gespeichert werden – entsprechend der ersten und letzten Sequenz aus L4, welche mit <1 3 2> beginnen. Für die neue Sequenz <2 1 3 2 6> wird, falls sie sich als groß erwiesen hat, die Nummer ihres zweiten Elternteils gespeichert – 1, für die Sequenz <2 1 3 2> die Nummer 3 – die Nummer der ersten neuen erhaltenen 5-Sequenz.

Wenn für eine Sequenz aus Lk-1 kein Nachfahre existiert, wird dieser Fakt in der Struktur der Nummern über eine ungültige Nummer gespeichert – beispielsweise -1.

2. Erste Variante

Der Prozeß der Arbeit des ersten Verfahrens kann in mehrere Etappen aufgegliedert werden:

(i) Suche aller großen 1-Sequenzen – der verschiedenen Elemente aus E – L1.

(ii) Suche aller großen 2-Sequenzen – L2 auf Grundlage von L1. Gleichzeitig wird für jede gefundene große Sequenz die Liste der Transaktionsnummern, welche diese enthalten, abgespeichert.

(iii) Suche der großen k-Sequenzen auf Grundlage von Lk-1. Die Etappe 3 wird solange wiederholt, bis $L_k = \varnothing$ erhalten wurde.

Die erste Variante des Verfahrens, für die Figur 3 schematisch ein Ablaufdiagramm zeigt, kann in Pseudocode wie folgt dargestellt werden:

```
L1 = {large 1-sequences};
C2 = candidate-gen(L1);              // 2-Kandidaten
forall transactions t ∈ D do begin
        Ct = subset(C2,t);     // Kandidaten, welche in t enthalten sind
        forall candidates c ∈ Ct do
```

```
                c.count++;
    end
    L2 = {c ∈ C2| c.count ≥ minsupp};
    for (k=3; Lk-1 ≠ ∅; k++) do begin
        Ck = candidate-gen(Lk-1);
        forall candidates c ∈ Ck do begin
            Tc = subset(c,D);       // Transaktionen, welche c enthalten
            c.support = |Tc|;
        end
        Lk = {c ∈ Ck | c.count ≥ minsupp};
    end
    answer = ULk;
```

In der ersten Variante des Verfahrens werden zwei Ansätze zur Auszählung der Supportwerte der Kandidaten-Sequenzen kombiniert. Zum Auffinden der Menge L2 wird der sequentielle Durchgang durch die Menge der Eingangstransaktionen genutzt. Für jede Transaktion werden die darin enthaltenen Kandidaten bestimmt, und für jeden Kandidaten wird der Zähler des Supportwertes um Eins erhöht.

In der ersten Variante des Verfahrens wurde in dieser Etappe der gleiche Ansatz gewählt wie in der nächsten Etappe: Für jeden Kandidaten wird die Liste der Transaktionen bestimmt, welche diesen Kandidaten enthalten können. Die Liste der Transaktionen ergibt sich als Schnittmenge über den gespeicherten Transaktionslisten der Eltern des betrachteten Kandidaten. Offensichtlich kann die Kandidaten-Sequenz nur in den Transaktionen enthalten sein, in denen gleichzeitig beide Elternteile dieses Kandidaten gespeichert sind. Die Liste der Transaktionen des Kandidaten enthält die Nummern der Transaktionen und unter Nutzung der Schnittmengenbildung werden dahin nur diejenigen Nummern hinzugefügt, welche in den Listen der Transaktionen seiner beiden Elternteile enthalten sind. Die derart erhaltene Liste erlaubt eine substantielle Reduktion der Anzahl der Transaktionen, welche auf das Vorhandensein eines bestimmten Kandidaten getestet werden müssen, besonders in den letzten Schritten des Verfahrens.

Jedoch setzt der Ansatz der Transaktionslisten den Durchlauf durch die Gesamtmenge der Kandidaten voraus. In der zweiten Etappe des Verfahrens, bei der Auszählung der Menge L2, sind nur die Elemente der Menge L1 bekannt. Ein beliebiges Paar großer 1-Sequenzen bildet einen Kandidaten der Länge 2. Während der Arbeit des Verfahrens mit realen Daten kann die Mächtigkeit der Menge L1 einige zehntausend 1-Sequenzen erreichen. Dann beträgt die Anzahl der Kandidaten der Länge 2 einige hundert Millionen. Eine vollständige Durchsicht aller

möglichen Kandidaten würde einen unvertretbar hohen Zeitaufwand erfordern. Es ist sinnvoller, in diesem Fall den Ansatz zu wählen, welcher vollständig in der zweiten Variante des Verfahrens benutzt wurde.

Gemäß der grundlegenden Idee der zweiten Variante des Verfahrens, wird für jeden Kandidaten ein Zähler seines Supportwertes eingeführt. Danach erfolgt der sequentielle Durchgang durch die Eingangsmenge der Transaktionen. Für jede Transaktion werden die in ihr enthalten Kandidaten bestimmt und ihre Zähler des Supports werden um Eins erhöht. In der zweiten Variante des Verfahrens wird ein komplexeres Schema zur Bestimmung der Kandidaten genutzt, welche in einer konkreten Transaktion enthalten sind, aber zur Bestimmung der Menge L2 kann eine einfachere Methode verwendet werden. Zur Bestimmung der Nummern aller Kandidaten sind zwei Durchläufe durch die Transaktionen notwendig, von denen der eine in den anderen eingebettet ist. Im äußeren Zyklus werden nacheinander die großen Elemente gewählt, welche in den Transaktionen enthalten sind; die nicht-großen Elemente werden ausgelassen. Der innere Zyklus beginnt mit dem Element, welches auf das in dem äußeren Zyklus ausgewählte Element folgt. Jedes ausgewählte Paar großer Elemente der Menge E stellt einen Kandidaten der Länge 2 dar und folglich ist es notwendig, den entsprechenden Zähler des Supportwertes des Kandidaten zu erhöhen. Die Nummer des Zählers wird ausgehend aus den Indexen der ausgewählten Elemente in der Liste der großen Elemente und der Gesamtzahl der gefundenen großen Elemente berechnet. Seien a und b – die entsprechend gewählten ersten und zweiten Elemente eines Kandidaten und N – die Anzahl der in der vorhergehenden Etappe des Verfahrens gefundenen 1-Sequenzen – der großen Elemente aus L1. Dann ergibt sich die Nummer des entsprechenden Kandidaten gemäß folgender Formel: $n_c = Na + b$.

Zum Ausschließen der wiederholten Erhöhung des Zählers des Supports eines speziellen Kandidaten bei der Behandlung ein und derselben Transaktion ist es notwendig, eine Methode zur Markierung des Faktes der Erhöhung des Zählers einer konkreten Transaktion vorzusehen. Die einfachste Methode besteht darin, für jeden Kandidaten außer seinem Zähler des Supports zusätzlich die Nummer der letzten Transaktion zu speichern, welche den Zähler des betrachteten Kandidaten inkrementiert hat. Da die Transaktionen aus der Eingangsmenge nacheinander behandelt werden, kann über den Inhalt des Feldes der Nummer der letzten Transaktion festgestellt werden, ob der Zähler seit Beginn der Behandlung der betrachteten konkreten Transaktion erhöht wurde.

Der Vorteil der beschriebenen Vorgehensweise zur Bestimmung der 2-Sequenzen L2 ist die relative Einfachheit der Berechnungen, da nur diejenigen Kandidaten behandelt werden, welche tatsächlich in den Eingangstransaktionen enthalten sind. Allerdings besitzt diese Methode auch einen gravierenden Nachteil. Für jeden Kandidaten müssen zwei mit ihm verbundene Werte gespeichert werden: der Zähler des Supportwertes und die Nummer der letzten, den Zähler inkrementierenden, Transaktion. Die Nutzung eines 32-bit Wertes für jedes Feld, welches theoretisch die Bearbeitung von maximal 2147483647 Transaktionen erlaubt, erfordert 8 Byte pro Kandidat. In der Etappe der Bestimmung der Menge L2 können Hunderte Millionen verschiedener Kandidaten existieren. Folglich benötigt diese Etappe der Arbeit des Verfahrens einige hundert Millionen Byte RAM-Speicher, was durchaus eine ernsthafte Einschränkung darstellt. Die quadratische Abhängigkeit des benötigten Speichers von der Anzahl der in der vorhergehenden Etappe gefundenen großen 1-Sequenzen stellt eine ernsthafte Einschränkung dar.

Um diese Beschränkung zu beseitigen, wurde in der endgültigen Form des ersten Verfahrens ein Schema des Mehrfachdurchlaufes der Bearbeitung der Transaktionsmenge im Verlauf der Konstruktion der Menge L2 realisiert. Während eines vollständigen Durchlaufes durch die Eingangsmenge der Transaktionen werden nur diejenigen Paare behandelt, deren erstes Element sich in einem bestimmten Wertebereich befindet. Auf diese Weise kann der Speicherplatz, welcher für die zweite Etappe des Verfahrens benötigt wird, unter Berücksichtigung des im Rechner verfügbaren Speicherplatzes beschränkt werden. Selbstverständlich werden mit stärkeren Speicherbeschränkungen umso mehr Durchläufe erforderlich und das Verfahren wird in der zweiten Etappe entsprechend langsamer. Daher stellt die beschriebene Methode der Bestimmung der 2-Sequenzen L2 einen Kompromiß zwischen benötigtem Speicherplatz und Bearbeitungsgeschwindigkeit dar. Diese Tatsache wird auch durch das experimentelle Laufzeitverhalten des Verfahrens bestätigt.

2.1 Weitere Details zur Realisierung der ersten Variante

In der ersten Etappe der Arbeit des Verfahrens müssen alle Elemente der Menge E gefunden werden, welche in den Eingangstransaktionen nicht weniger als eine vorgegebene Anzahl auftauchen. Da die Nummern und Anzahlen der verschiedenen Elemente aus E anfänglich nicht bekannt sind, ist es notwendig, eine Methode zur Speicherung des eigentlichen Elementes sowie der Anzahl der Transaktionen, in der dieses enthalten ist, vorzusehen. In der

Realisierung beider Varianten des Verfahrens erfolgt ein Durchlauf durch alle Eingangstransaktionen. Zur Speicherung der Elemente und ihrer Zähler wird eine Standardklasse (z.B. aus Java) herangezogen – die *Hashtable*. Diese Klasse erlaubt die Speicherung des Paares <Schlüssel, Wert>, welche zur schnellen Suche der Werte den Mechanismus der Hash-Tabelle bezüglich der Schlüsselwerte benutzt. In der Realisierung des Verfahrens wird als Schlüssel ein Objekt der Klasse (z.B. aus Java) *Integer* (Wrapper-Klasse für Integer-Werte) benutzt, welche mit dem Wert des gewählten Elementes initialisiert wird, und als Wert – ein anderes Objekt der Klasse *Integer*, welches mit dem Wert seines Zählers initialisiert wird. Bei der Auswahl eines weiteren Elementes einer Transaktion erfolgt ein Zugriff zum Objekt der Klasse *Hashtable*, um den Wert über den Schlüssel zu erhalten, welcher gleich dem gewählten Element ist. Wenn ein solcher Wert in der Hash-Tabelle gefunden wurde, so wird es erneut, um Eins erhöht, zurückgeschrieben. Wenn kein solcher Wert für den Schlüssel gefunden wurde, so wird in die Hash-Tabelle das Paar eingetragen, welches als Wert den Anfangswert des Zählers unter Berücksichtigung des bereits gewählten Elementes enthält – 1.

Zur Vermeidung wiederholter Inkrementierungen des Zählers einer Transaktion wird in der Realisierung des Verfahrens eine andere Klasse benutzt (z.B. aus Java) – *Vector*, in der alle bereits von einer Transaktion bearbeiteten Elemente gespeichert werden. Bevor es den Wert in der Hash-Tabelle sucht, versucht das Verfahren das Element in der Instanz der Klasse *Vector* zu finden. Wenn ein solches Element gefunden wurde, so bedeutet dies, dass es bereits in dieser Transaktion bearbeitet wurde und es findet ein Übergang zum nächsten Element der Transaktion statt. Natürlich wird der Vektor der Werte beim Übergang des Verfahrens zur nächsten Transaktion gesäubert. Nach dem Durchgang durch die Transaktionsmenge werden nur jene Elemente selektiert, deren Zählerwerte nicht kleiner als der Minimalwert sind. Die gefundenen großen Elemente werden im Array abgespeichert und werden sortiert, um im weiteren Verfahren die Binärsuche der Elemente anwenden und auf die Folge der ganzen positiven Zahlen 0, ..., N-1 abbilden zu können, wobei N die Anzahl der gefundenen großen Elemente ist. Aus den großen Elementen wird die Menge L1 der großen 1-Sequenzen formiert.

Zur Speicherung der Liste der Nummern der Transaktionen wird in der Realisierung des Verfahrens die Klasse *TransactionList* benutzt. Sie erlaubt die Speicherung der Nummern der Transaktionen in einem Array, das Hinzufügen einer Nummer an das Ende der Liste, wobei vorausgesetzt wird, dass die Nummern in aufsteigender Reihenfolge hinzugefügt werden.

Diese Voraussetzung ist unabdingbar für die korrekte Arbeit der Methode der Klasse *TransacionList*, welche die Schnittmengenbildung beider Transaktionslisten realisiert. Die Anfangslisten der Transaktionen werden im Verlauf des sequentiellen Durchlaufes durch die Menge der Eingangstransaktionen in der Etappe der Konstruktion von L2 gebildet, hierbei werden in natürlicher Weise die Bedingungen der Sortierordnung sichergestellt. Im weiteren entstehen die neuen Listen während der Ausführung der Schnittmengenoperation. Die Listen der Nummern werden in einem Array gespeichert, aber da die Länge des Arrays in der Etappe der dynamischen Speicherallokation bestimmt werden, ist es notwendig, im erforderlichen Maße zusätzliche Felder des Arrays bereitzustellen, indem seine Länge erhöht wird. Während der Erzeugung einer Instanz der Klasse *TransactionList* wird ein Array mit einer Anfangsgröße angelegt und schrittweise gefüllt. Bei Erreichen der Maximalanzahl der Nummern wird der Speicher erneut bereitgestellt. Die maximale Speicherkapazität des Arrays wird beispielsweise um das Zehnfache erhöht, es wird ein neues Array der gleichen maximalen Länge bereitgestellt und darin wird der Inhalt des vorhergehenden Arrays kopiert. Das alte Array kann als überflüssig markiert werden (z.B. für die Virtual Machine in Java).

In der dritten Etappe der Arbeit des Verfahrens werden alle im vorhergehenden Schritt gespeicherten großen k-Sequenzen durchgegangen. Über die gespeicherte Nummer des zweiten Elternteils der Sequenz werden, unter Nutzung der oben beschriebenen Datenstruktur, die Nummern der ersten und letzten großen k-Sequenzen gefunden, mit denen die gefundene Sequenz vereinigt werden kann. Weiter wird die Schnittbildungsoperation über den Listen der Transaktionen beider Sequenzen ausgeführt. Wenn die Länge der resultierenden Liste kleiner als der notwendige minimale Supportwert ist, so wird der betrachtete Kandidat sofort verworfen, sogar ohne Operation eines unmittelbaren Kandidaten und der Auszählung der Anzahl der Transaktionen, welche diesen enthalten. Andernfalls erfolgt die Erzeugung der Sequenz der Elemente des Kandidaten selbst und die Überprüfung jeder Transaktion aus der Schnittbildung erhaltenen Liste auf den Inhalt der in ihr enthaltenen Kandidaten-Sequenzen. Wenn in einer Transaktion eine entsprechende Sequenz gefunden wurde, so wird die Nummer dieser Transaktion gespeichert. Auf diese Weise liegt nach der Bearbeitung eines Kandidaten eine Liste der Transaktionen vor, welche den Kandidaten enthalten. Wenn die Anzahl solcher Transaktionen nicht kleiner der Mindestanzahl ist, so wird dieser Kandidat zusammen mit der Nummer seines zweiten Elternteils gespeichert.

Die oben beschriebenen Aktionen werden für jeden Kandidaten wiederholt. Wenn im Ergebnis keine einzige große Sequenz gefunden wurde, so wird die Arbeit der dritten Etappe des Verfahrens beendet und die Ergebnisse werden über das Interface ausgegeben. Es kann zwei Gründe geben, warum im Verlauf eines Schrittes keine einzige große Sequenz gefunden wurde. In der Menge Lk kann keine Sequenz vorliegen, welche einen Kandidaten mit einer anderen Sequenz bilden könnte, dass heißt, in einem Schritt existiert kein einziger Kandidat. Ein anderer Grund kann die Variante sein, in der im Ergebnis kein einziger Kandidat eine große Sequenz darstellt.

Zusammenfassend erweist sich die erste Variante des Verfahrens als schnell auf großen Datenmengen und weist einen sparsamen Speicherplatzbedarf auf. Bezüglich Speicherplatz und Rechenzeit hinsichtlich der Benutzung von Transaktionslisten zeigen empirische Berechnungen und experimentelle Resultate, dass auf realen Daten ihre Größe eine Tendenz zur starken Verringerung aufweist, da mit Erhöhung der Länge der bearbeiteten Sequenzen diese in einer zunehmend geringeren Anzahl von Transaktionen enthalten sind und sich folglich die Länge der Listen verringert wie auch die zu ihrer Bearbeitung benötigte Zeit. Folglich muß für die Arbeit der ersten Variante des Verfahrens die Eingangsmenge der Transaktionen in Form einer Liste von Sequenzen ganzer Zahlen eingegeben werden, der minimale Supportwert als Floatwert und ferner der verfügbare Speicherplatz für die zweite Etappe des Verfahrens.

3. Zweite Variante

Die zweite Variante des Verfahrens, für die Figur 4 schematisch ein Ablaufdiagramm zeigt, wurde zum Zweck des Vergleichs mit der ersten Variante des Verfahrens bezüglich Geschwindigkeit und Speicherplatz erarbeitet und realisiert. Die zweite Variante kann in Pseudocode wie folgt dargestellt werden:

```
L1 = {large 1-sequences};
for (k=2; Lk-2 ≠ ∅; k++) do begin
    Ck = candidate-gen(Lk-1);    // neue Kandidaten
    forall transactions t ∈ D do begin
        Ct = subset(Ck, t);
        forall candidates c ∈ Ct do
            c.count++;
    end
    Lk = {c ∈ Ck | c.count ≥ minsup}
end
answer = U Lk;
```

Das dargestellte Verfahren ist sehr allgemein und konkretisiert nicht die Methode der Auswahl der Kandidaten, welche in einer konkreten Transaktion enthalten sind. Im einfachsten Fall kann dies die Durchsicht aller Kandidaten sein mit dem Ziel, diejenigen auszuwählen, welche in der Sequenz der Transaktion enthalten sind. Eine andere Variante ist die Durchsicht aller Sequenzen einer gegebenen Länge und die Auswahl derjenigen aus ihnen, welche Kandidaten darstellen. Es ist klar, dass beide Methoden wegen deren geringer Effizienz nachteilig sind; in der Realisierung der zweiten Variante wird daher eine andere Methode beschrieben.

Die zweite Variante benutzt eine Reihe gleicher Ideen wie die erste Variante. Insbesondere wird zur Berechnung der Menge L1 der gleiche Ansatz ohne große Änderungen benutzt. Die gesamte Beschreibung der ersten Etappe der Arbeit der ersten Variante des Verfahrens, welche oben angeführt wurde, gilt auch bezüglich der zweiten Variante. Wie bereits oben erwähnt, benutzt die zweite Variante des Verfahrens zur Summierung des Supports der Kandidaten die Idee, welche in der ersten Variante nur in der zweiten Etappe zur Generierung der Menge der großen 2-Sequenzen L2 benutzt wurde.

3.1 Weitere Details zur Realisierung der zweiten Variante

Um für jede Transaktion die Möglichkeit zur Bestimmung der Liste der in ihr enthaltenen Kandidaten zu erhalten, wird in der zweiten Variante des Verfahrens eine spezielle Datenstruktur benutzt. Beginnend mit der zweiten Etappe des Verfahrens wird jede Transaktion als Liste der Nummern der im vorhergehenden Schritt des Verfahrens gefundenen k-Sequenzen dargestellt. Unter Durchsicht der Paare dieser k-Sequenzen und deren Zusammenfügung gemäß der oben beschriebenen Methode der Kandidatengenerierung kann man alle Kandidaten der Länge k+1 erhalten, welche in einer bestimmten Transaktion enthalten sind. Jedoch ist zur Trennung der Kandidaten die Liste der großen k-Sequenzen als zweidimensionales Array von Werten organisiert. Die resultierende Datenstruktur wird im nächsten Beispiel illustriert:

Angenommen, die Eingangsmenge der Transaktionen enthält folgende Transaktion:

<3 1 3 2 1>

Es wird weiterhin angenommen, dass in der zweiten Etappe der Arbeit des Verfahrens festgestellt wurde, dass die in dieser Transaktion enthaltenen Sequenzen <3 1>, <3 2>, <1 2> und <1 1> groß sind und jeweils die Nummern 5, 7, 3 und 1 besitzen. Dann wurde im Ergebnis

der zweiten Etappe der Arbeit des Verfahrens folgende Datenstruktur erzeugt, welche der gegebenen Transaktion entspricht:

1. 5 7
2. 3 1
3. 7 5
4. –

Die Anzahl der Zeilen in dieser Struktur ist gleich der Anzahl der Elemente in der Transaktion während der Etappe der Konstruktion der Menge L2. In den weiteren Etappen ist diese Menge gleich der Zahl der Zeilen in der vorhergehenden Struktur für diese Transaktion. Jede Zeile enthält die Nummern der großen Sequenzen, welche mit diesem Element beginnen oder mit dieser Zeile in der Struktur, welche im vorhergehenden Schritt erzeugt wurde. In jedem Schritt werden die Paare der Nummern der großen Sequenzen des vorhergehenden Levels betrachtet und das Verfahren durchsucht alle Paare von Nummern, welche vereinigt werden können. Der Test auf die Möglichkeit zur Verknüpfung wird auf Basis des gleichen Prinzips wie in der ersten Variante des Verfahrens ausgeführt.

Eine derartige Struktur wird für jede Transaktion gebildet und wird in jedem Schritt des Verfahrens aktualisiert. Die Erststruktur wird in der zweiten Etappe des Verfahrens erzeugt und das ist deren einziger Unterschied zur entsprechenden Etappe der ersten Variante des Verfahrens. In einem Schritt des Verfahrens werden nacheinander alle Transaktionen der Eingangsmenge behandelt, wobei die beschriebene Struktur genutzt wird. Schrittweise wird jedes Element jeder Zeile gewählt und danach werden die Elemente jeder nachfolgenden Zeile durchsucht. Wenn das derart gewählte Paar einen Kandidaten darstellen kann, so wird der Zähler seines Supports um Eins erhöht. Zur Verhinderung einer mehrfachen Inkrementierung des Zählers während der Bearbeitung einer Transaktion kann die Idee der ersten Variante des Verfahrens herangezogen werden – für jeden Kandidaten wird die Nummer der letzten inkrementierenden Transaktion gespeichert.

Auf diese Weise zeigt sich, dass in der zweiten Variante des Verfahrens zur Laufzeit gar keine reelle Generierung von Kandidaten-Sequenzen vonnöten ist, da dies nicht für die Summierung der Supportwerte benötigt wird – im Gegensatz zum ersten Verfahren. Die im k-ten Schritt gefundenen großen k-Sequenzen werden in Form der Paare ihrer Eltern gespeichert. Dadurch wird Speicherplatz gespart, da in der Praxis die großen k-Sequenzen am Ende der eigentlichen Arbeit des Verfahrens entstehen können, während des Durchgangs von der Men-

ge L1 zur Menge Ln und keinen überflüssigen Speicherplatz während der Arbeit des Hauptzyklus des Verfahrens einnehmen.

Schwierig kann sich der Prozeß des Füllens der benötigten Struktur gestalten, da während der Bearbeitung der Transaktion die Nummern der Kandidaten bekannt sind, nicht aber die der großen Sequenzen, welche erst nach der Bearbeitung aller Transaktionen bestimmt werden. Somit werden während der Bearbeitung in die erneuerte Struktur die Nummern der in ihnen enthaltenen Kandidaten eingetragen. Nach Bearbeitung aller Transaktionen der Eingangsmenge müssen die in der Struktur enthaltenen Nummern unter Berücksichtigung der Kandidaten, welche die Supportbedingung nicht erfüllten, korrigiert werden. Der Teil des Verfahrens, in welchem die Nummern korrigiert werden, kann wie folgt dargestellt werden:

```
m = 0; // Akkumulator der Korrektur
forall c∈ Ck do begin                    // für jeden Kandidaten
        if c.count>=minsup then c.correction = -m-1
        else m++;
end
```

Nach Ausführung dieses Fragments des Verfahrens ist für jeden Kandidaten der Wert der Korrektur bekannt, welcher von der Nummer des Kandidaten abgezogen werden muß, um die Nummer der großen Sequenz zu erhalten. In der Tat, um die Nummer der Sequenz aus der Nummer des Kandidaten zu erhalten, muß die Anzahl der den Test nicht bestandenen Kandidaten abgezogen werden, deren Nummern kleiner der Nummer des betrachteten Kandidaten sind. Die letzte Etappe der Erneuerung der Struktur der Transaktionen ist die Transformation der Nummern der Kandidaten in die Nummern der großen Sequenzen, gleichzeitig mit der Entfernung der Nummern der nicht-großen Kandidaten und der Sortierung.

Die Sortierung dient der Beschleunigung der Suche der paarweisen Sequenzen während der Bearbeitung der Transaktionsstrukturen, dass heißt, derjenigen Sequenzen, mit denen die gewählte Sequenz vereinigt werden kann. Ohne die Sortierung der Zeilen der Strukturen der Transaktion würde die Suche des zweiten Elternteils einen vollständigen Durchgang aller nachfolgenden Zeilen mit Prüfung der Bedingung der Vereinigung bedeuten. Mit Hilfe der Sortierung der Werte der Nummern der großen Sequenzen wird eine steigende Ordnung eingeführt und folglich ergibt sich die Möglichkeit, mittels Binärsuche in schnellster Weise einen Wert in der Zeile zu finden, mit welchem beginnend die Nummern der Eltern-Sequenzen gespeichert werden. Weiter wird ein geordneter Durchgang durch die Zeile ausgeführt, bis eine

Nummer einer Sequenz größer der maximal möglichen Elternnummer für den gewählten ersten Elternteil gefunden wird. Experimentelle Daten haben gezeigt, dass die eingeführte Sortierung, welche nicht unbedingt für die Arbeit des Verfahrens notwendig ist, die Arbeit in den Fällen beschleunigt, in denen in der mit einer gewissen Transaktion verbundenen Struktur nacheinander einige sehr lange Zeilen enthalten sind. In der ersten Variante, ohne Sortierung, konnte ein deutlicher Abfall der Effektivität in einigen Teilen der Transaktionsmenge beobachtet werden, weswegen die Gesamtzeit der Arbeit des Verfahrens stark verlängert wird. Nach der Realisierung der Sortierung arbeitete das Verfahren auf solchen Daten um ein Mehrfaches schneller, abhängig von den konkreten Eingangstransaktionsmengen.

Während der Bearbeitung der Transaktionsstrukturen werden nacheinander die Nummern der großen Sequenzen ausgewählt, welche im vorhergehenden Schritt des Verfahrens gefunden worden sind und welche in der laufenden bearbeiteten Transaktion enthalten sind. Für jede gewählte Nummer entsteht die Aufgabe der Bestimmung der Nummern des ersten und des letzten Elternteils, mit denen die Sequenz mit der gewählten Nummer vereinigt werden kann. Außerdem muß für jeden Kandidaten die Nummer bekannt sein, um den Zähler des Supports erhöhen zu können. Teilweise wird diese Aufgabe wie in der ersten Variante des Verfahrens gelöst. Jedoch wird in der Realisierung der zweiten Variante eine leicht abgewandelte Idee benutzt, unter Berücksichtigung der Notwendigkeit, die Nummer des Kandidaten zu kennen. In der ersten Variante ist die Nummer des Kandidaten bekannt, da im Hauptzyklus faktisch ein Durchlauf durch die Kandidatenmenge stattfindet und zur schnellen Ermittlung der ersten und letzten Nummer möglicher Eltern wird in jedem Schritt eine spezielle Struktur erneuert. In der zweiten Variante wird eine modifizierte Struktur angewandt, unter Berücksichtigung der Notwendigkeit der Speicherung der Nummern der entsprechenden Kandidaten. Außer den Nummern möglicher Eltern wird in jedem Eintrag dieser Struktur gleichfalls die Nummer des ersten Kandidaten, welcher erzeugt werden kann, gespeichert. Auf diese Weise können über die Nummer des zweiten Elternteils einer großen Sequenz gleich die Nummern der Eltern ermittelt werden, mit welchen die betrachtete Sequenz vereinigt werden kann, sowie die Nummern der entsprechenden Kandidaten.

Die zweite Variante des Verfahrens erfordert, wie auch die zweite Etappe des ersten Verfahrens, die Allokation von Speicher für die Werte der Zähler des Supports und das Array der Nummern der Transaktion, welche als letzte den Zähler inkrementierte.

4. Beschreibung eines Programminterfaces

Das Programminterface wird in Form eines Beispiels der Benutzung des Verfahrens in einem Java-Programm betrachtet.

Die Klasse *TransactionSetSeq* wird zur Speicherung des Satzes der sequentiellen Eingangstransformationen benutzt. Eine Instanz der Klasse wird mittels folgender Anweisung realisiert:

TransactionSetSeq tss = new TransactionSetSeq();

Nun fügen wir alle Transaktionen des Eingangssatzes hinzu:

for (int i=0;i< <Anzahl Transaktionen>;i++) {

Jede sequentielle Transaktion wird als eine Instanz der Klasse *ItemSetSeq* repräsentiert:

ItemSetSeq iss = new ItemSetSeq();

Die Elemente der Transaktionen sind ganze Zahlen – zum Hinzufügen eines neuen Elementes zur Transaktion dient folgende Methode der Klasse *ItemSetSeq*:

void addItem(int item);

Wir fügen alle Elemente der aktuellen Transaktion hinzu:

for (int j=0;j< <Anzahl der Elemente der aktuellen Transaktion>; j++)

iss.addItem(<Nummer des Elementes>);

Zum Hinzufügen einer Transaktion zum Transaktionssatz wird folgende Methode der Klasse *TransactionSetSeq* benutzt:

void addTransaction(ItemSetSeq iss);

tss.addTransaction(iss);

}

Zur Berechnung der Sequenzen muß eine Instanz der Klasse *Sequential* oder *Sequential2* erzeugt werden:

Sequential seq = new Sequential();

Sequential2 seq2 = new Sequential2();

Zum Starten des Verfahrens ist es notwendig, folgende in den Klassen *Sequential* und *Sequential2* enthaltene Methode aufzurufen:

SequentialResult SequentialAlg(TransactionSetSeq tss, double minsupp);

wobei

tss – Eingangsmenge der Transaktionen,

minsupp – minimaler Supportwert.

SequentialResult seqResult = seq.SequentialAlg(tss,minsupp);

oder

SequentialResult seqResult = seq2.SequentialAlg(tss,minsupp);

Das Ergebnis der Arbeit des Verfahrens wird in dem Objekt der Klasse *SequentialResult* gespeichert. Zum Zugriff der gefundenen Sequenzen enthält die Klasse *SequentialResult* folgende Methode:

ItemSetSeqList getLargeSequences();

ItemSetSeqList issList = seqResult.getLargeSequences();

Die Klasse *ItemSetSeqList* enthält die Methoden:

int getSize(); – liefert die Anzahl der gefundenen Sequenzen,

ItemSetSeq getItemSetAt(int index); – liefert die Sequenz der Nummer index.

for (int i=0;i < issList.getSize(); i++) {

ItemSetSeq iss = issList.getItemSetAt(i);

Zum Zugriff der Sequenz enthält die Klasse *ItemSetSeq* folgende Methode:

int getItemAt(int index);

Folgende Methode liefert die Länge der Sequenz:

int getSize();

```
for (int j=0;j<iss.getSize();j++)
    int item = iss.getItemAt(j);
}
```

Den Wert des Zählers für den Support liefert folgende Methode der Klasse *ItemSetSeq*:

```
int getSupportCount();
```

Der eigentliche Supportwert *support* für jede Sequenz ist der Quotient des Wertes des Support-Zählers und der Anzahl aller Transaktionen.

5. Beispiel: Clickstreamanalyse

Im weiteren wird anhand der Figuren 1 und 2 die Anwendung des beschriebenen Verfahrens am Beispiel einer Clickstreamanalyse von Webservern erläutert.

Um über das Internet eine Website aufzurufen, benötigt der Nutzer einen Web-Browser 10, z.B. Netscape Navigator, mit dem er sich mit einem Web-Server 20, z.B. dem Apache Web-Server, verbindet. Webserver speichern die Informationen über abgerufen HTML-Seiten einer Website in Datenbanken 30 oder Logfiles 40. Die Information über die abgerufenen HTML-Seiten wird hierbei in einer Tabelle gespeichert. Die Spalten entsprechen den Informationen über die abgerufene Seite wie beispielsweise Datum und Uhrzeit, URL der aufgerufenen HTML-Seite, Referer-URL, Typ des Web-Browsers. Die Zeilen entsprechen den abgerufenen HTML-Seiten selbst. Über verschiedene Mechanismen, die hier nicht weiter ausgeführt werden sollen, ist es möglich, Websessions zu identifizieren und jeder abgerufenen Seite zuzuordnen. Eine Websession ist eine Sequenz der von einem User der Website hintereinander abgerufenen HTML-Seiten.

Ziel der Clickstreamanalyse ist das Herausfinden typischer Pfade, entlang derer sich die User einer Website hauptsächlich bewegen. Beispielsweise kann es für einen E-Commerce Shop passieren, dass viele Nutzer über die Hauptseite den Shop betreten, in eine gewisse Produktkategorie gehen, ein spezielles Produkt in Detailansicht betrachten und danach - den Shop verlassen. Das lässt darauf schließen, dass das betrachtete Produkt die Erwartungen verfehlte und seine Präsentation dringend verbessert werden muß. Wenn Nutzer zyklische Pfade wählen, lässt sich oftmals daraus schließen, dass sie der Shop unübersichtlich aufgebaut ist, usw.

Mit Hilfe eines Datenzugriffs-Moduls 50 greift nun eine mittels eines Programms implementierte Data-Mining-Anwendung 60, welche die eigentliche Clickstreamanalyse ausführt, auf die Daten der abgerufenen HTML-Seiten zu, welche sich entweder in der Datenbank 30 oder in Logfiles 40 befinden. Hierbei führt das Datenzugriffs-Modul 50 verschiedene Transformationen aus, wie beispielsweise das Auflösen von IP-Adressen, das Erkennen von Datumsformaten sowie die oben beschriebene Extraktion von Websessions.

Die Mining-Anwendung 60 berechnet nun die häufigsten Pfade der Nutzer durch die Website. Das Herausfinden der typischen Pfade in Websites stellt eine Aufgabe der Sequenzanalyse dar. Hierbei ist E = {e1, e2, ...,em} die Menge aller HTML-Seiten einer Website. Jede Transaktion di entspricht einer Websession, deren Elemente eine Sequenz $<s_0, s_1, \ldots, s_k>$ von HTML-Seiten bilden. Die Gesamtzahl der Transaktionen D entspricht allen Websessions des Webservers. Für ein einfaches Beispiel sind die Clickstream-Transaktionen in 80 bzw. (nach Abbildungen 90 und 100 in den Bereich der ganzen Zahlen) in 110 dargestellt. Die Aufgabe der Clickstreamanalyse besteht nun darin, für einen vorgegebenen minimalen Supportparameter s alle großen Sequenzen von HTML-Seiten zu finden, welche die Ergebnisse 70 der Mining-Analyse bilden. Für das Beispiel der Clickstream-Tabelle 80 sind die gefundenen großen Sequenzen in 120 bzw. (nach Rückabbildung) in 130 für einen minimalen Supportwert von 50 % angegeben. Diese stellen die typischen Nutzerpfade der Website dar, deren Analyse - wie oben dargelegt - zu einer Verbesserung der Struktur der Website genutzt werden kann. Insbesondere können die Ergebnisse direkt in den Webserver eingebaut werden, beispielsweise in Form von empfohlenen HTML-Seiten (Recommender-Systeme).

Zur Berechnung der großen Sequenzen ist das in der Erfindung beschriebene Verfahren sehr gut geeignet, insbesondere deswegen, weil es große Datenmengen analysieren kann. Gerade Logfile-Tabellen stark besuchter Websites weisen aber häufig Millionen abgerufener HTML-Seiten auf und stellen somit höchste Ansprüche an Sequenzanalyseverfahren. Die Auswertung dieser Informationen mittels Sequenzanalyse war daher zumeist außerordentlich rechenzeitintensiv und auf Sonderfälle (Sequenzen der Länge 2, nur wenige Websessions, usw.) beschränkt. Das beschriebene Verfahren überwindet diese Beschränkung für nahezu alle auf derzeitigen Webservern anfallenden Clickstream-Daten.

Die in der vorstehenden Beschreibung, den Ansprüchen und der Zeichnung offenbarten Merkmale der Erfindung können sowohl einzeln als auch in beliebiger Kombination für die

Verwirklichung der Erfindung in ihren verschiedenen Ausführungsformen von Bedeutung sein.

Ansprüche

1. Verfahren zum Ermitteln einer Menge großer Sequenzen aus einer elektronischen Datenbank mit einer Folge D={d1, ..., dn} von Transaktionen di ($1 \leq i \leq n$) in einem Computersystem mit einem implementierten Abfragemodul, wobei jede der großen Sequenzen auf der Folge D von Transaktionen di einen Stützwert aufweist, der größer oder gleich einem vorgegebenen Stützwert S ist, wobei jede der Transaktionen di der Folge D eine Sequenz von Elementen eines Satzes E={e1, ..., em} von Elementen ej ($1 \leq j \leq m$) ist und wobei das Verfahren die folgenden Schritte umfaßt:
   a) Ermitteln einer Menge L1 großer Sequenzen aus der Folge D von Transaktionen, wobei die großen Sequenzen der Menge L1 jeweils genau ein Element des Satzes E umfassen und ein zugehöriger Stützwert S_{L1} auf der Folge D von Transaktionen jeweils größer oder gleich dem vorgegebenen Stützwert S ist;
   b) Ermitteln einer Menge L2 großer Sequenzen aus der Folge D von Transaktionen, wobei die großen Sequenzen der Menge L2 jeweils genau zwei Elemente des Satzes E in einer jeweiligen Reihenfolge R_{L2} umfassen und ein zugehöriger Stützwert S_{L2} auf der Folge D von Transaktionen jeweils größer oder gleich dem vorgegebenen Stützwert S ist und wobei beim Ermitteln der Menge L2 großer Sequenzen nur Sequenzen berücksichtigt werden, die als Teilsequenz eine der großen Sequenzen der Menge L1 umfassen;
   c) Ermitteln einer Menge Lk ($k > 2$) großer Sequenzen aus der Folge D von Transaktionen, wobei die großen Sequenzen der Menge Lk jeweils genau k Elemente des Satzes E in einer jeweiligen Reihenfolge R_{Lk} umfassen und ein zugehöriger Stützwert S_{Lk} auf der Folge D von Transaktionen jeweils größer oder gleich dem vorgegebenen Stützwert S ist, wobei beim Ermitteln der Menge Lk nur Sequenzen berücksichtigt werden, die als sich teilweise überlappende Teilsequenzen zwei der großen Sequenzen der Menge Lk-1 mit der jeweiligen Reihenfolge R_{Lk-1} umfassen; und
   d) Wiederholen des Schrittes c) für k=k+1 und Abbrechen des Wiederholens des Schrittes c) beim Erfüllen einer vorgegebenen Abbruchbedingung.

2. Verfahren nach Anspruch 1, wobei beim Ermitteln der Menge L2 im Schritt b) bzw. der Menge Lk im Schritt c) die Folge D von Transaktionen nach Kandidaten-Sequenzen durchsucht wird, die als sich teilweise überlappende Teilsequenzen zwei der großen Se-

quenzen der Menge L1 bzw. der Menge Lk-1 umfassen, und wobei für jede erstmalig gefundene Kandidaten-Sequenz ein zugehöriger Stützwert-Zähler registriert wird und der zugehörige Stützwert-Zähler erhöht wird, wenn die jeweilige Kandidaten-Sequenz beim Durchsuchen der Folge D von Transaktionen erneut ermittelt wird.

3. Verfahren nach Anspruch 1, wobei:
   - beim Ermitteln der Menge L2 im Schritt b) die Folge D von Transaktionen nach Kandidaten-Sequenzen durchsucht wird, die als Teilsequenz eine der großen Sequenzen der Menge L1 umfassen, und für jede erstmalig gefundene Kandidaten-Sequenz ein zugehöriger Stützwert-Zähler registriert wird und der zugehörige Stützwert-Zähler erhöht wird, wenn die jeweilige Kandidaten-Sequenz beim Durchsuchen der Folge D von Transaktionen erneut ermittelt wird; und
   - beim Ermitteln der Menge Lk ($k > 2$) im Schritt c) eine Kandidatenmenge von Sequenzen gebildet wird, die sämtliche paarweise Kombinationen der Sequenzen der Menge Lk-1 umfaßt, aus der Folge D von Transaktionen die Transaktionen bestimmt werden, die wenigstens eine Sequenz der Kandidatenmenge umfassen, und der Stützwert für die Sequenzen der Kandidatenmenge ermittelt wird, für die eine diese Sequenz umfassende Transaktion gefunden wurde.

4. Verfahren nach Anspruch 2 oder 3, wobei für jeden zugehörigen Stützwert-Zähler ein Anzeigewert erzeugt und aktualisiert wird, der anzeigt, für welche Transaktion der zugehörige Stützwert-Zähler zuletzt erhöht wurde.

5. Verfahren nach Anspruch 3 oder 4, wobei aus der Kandidatenmenge von Sequenzen, die sämtlichen paarweisen Kombinationen der Sequenzen der Menge L1 entsprechen, nur Sequenzen berücksichtigt werden, deren erstes Element in einem vorgegebenem Wertebereich liegt.

6. Computerprogramm-Produkt zum Ermitteln einer Menge großer Sequenzen aus einer elektronischen Datenbank mit einer Folge $D=\{d1, ..., dn\}$ von Transaktionen di ($1 \leq i \leq n$) in einem Computersystem mit einem implementierten Abfragemodul, wobei jede der großen Sequenzen auf der Folge D von Transaktionen di einen Stützwert aufweist, der größer oder gleich einem vorgegebenen Stützwert S ist, wobei jede der Transaktionen di

der Folge D eine Sequenz von Elementen eines Satzes E={e1, ..., em} von Elementen ej ($1 \leq j \leq m$) ist und wobei das Produkt die folgenden Mittel umfaßt:

a) auf einem elektronischen Speichermedium aufgezeichnete Mittel zum Ermitteln einer Menge L1 großer Sequenzen aus der Folge D von Transaktionen, wobei die großen Sequenzen der Menge L1 jeweils genau ein Element des Satzes E umfassen und ein zugehöriger Stützwert S_{L1} auf der Folge D von Transaktionen jeweils größer oder gleich dem vorgegebenen Stützwert S ist;

b) auf dem elektronischen Speichermedium aufgezeichnete Mittel zum Ermitteln einer Menge L2 großer Sequenzen aus der Folge D von Transaktionen, wobei die großen Sequenzen der Menge L2 jeweils genau zwei Elemente des Satzes E in einer jeweiligen Reihenfolge R_{L2} umfassen und ein zugehöriger Stützwert S_{L2} auf der Folge D von Transaktionen jeweils größer oder gleich dem vorgegebenen Stützwert S ist und wobei beim Ermitteln der Menge L2 großer Sequenzen nur Sequenzen berücksichtigt werden, die als Teilsequenz eine der großen Sequenzen der Menge L1 umfassen;

c) auf dem Speichermedium aufgezeichnete Mittel zum Ermitteln einer Menge Lk (k > 2) großer Sequenzen aus der Folge D von Transaktionen, wobei die großen Sequenzen der Menge Lk jeweils genau k Elemente des Satzes E in einer jeweiligen Reihenfolge R_{Lk} umfassen und ein zugehöriger Stützwert S_{Lk} auf der Folge D von Transaktionen jeweils größer oder gleich dem vorgegebenen Stützwert S ist, wobei beim Ermitteln der Menge Lk nur Sequenzen berücksichtigt werden, die als sich teilweise überlappende Teilsequenzen zwei der großen Sequenzen der Menge Lk-1 mit der jeweiligen Reihenfolge R_{Lk-1} umfassen; und

d) auf dem elektronischen Speichermedium aufgezeichnete Mittel zum Wiederholen des Schrittes c) für k=k+1 und Abbrechen des Wiederholens des Schrittes c) beim Erfüllen einer vorgegebenen Abbruchbedingung.

7. Integriertes Sequenzanalysesystem mit:

- einer elektronischen Datenbank mit einer Folge D={d1, ..., dn} von Transaktionen di ($1 \leq i \leq n$), wobei jede der großen Sequenzen auf der Folge D von Transaktionen di einen Stützwert aufweist, der größer oder gleich einem vorgegebenen Stützwert S ist, wobei jede der Transaktionen di der Folge D eine Sequenz von Elementen eines Satzes E={e1, ..., em} von Elementen ej ($1 \leq j \leq m$) ist;

- einem Abfragemodul, welches eine mit der Datenbank gekoppelte Abfrageeinrichtung und eine Verarbeitungseinrichtung zum Erfassen von Abfrageparametern und zum Erzeugen von Abfragen an die Abfrageeinrichtung umfaßt;
- Mitteln zum Ermitteln einer Menge L1 großer Sequenzen aus der Folge D von Transaktionen, wobei die großen Sequenzen der Menge L1 jeweils genau ein Element des Satzes E umfassen und ein zugehöriger Stützwert S_{L1} auf der Folge D von Transaktionen jeweils größer oder gleich dem vorgegebenen Stützwert S ist;
- Mitteln zum Ermitteln einer Menge L2 großer Sequenzen aus der Folge D von Transaktionen, wobei die großen Sequenzen der Menge L2 jeweils genau zwei Elemente des Satzes E in einer jeweiligen Reihenfolge R_{L2} umfassen und ein zugehöriger Stützwert S_{L2} auf der Folge D von Transaktionen jeweils größer oder gleich dem vorgegebenen Stützwert S ist und wobei beim Ermitteln der Menge L2 großer Sequenzen nur Sequenzen berücksichtigt werden, die als Teilsequenz eine der großen Sequenzen der Menge L1 umfassen;
- Mitteln zum Ermitteln einer Menge Lk ($k > 2$) großer Sequenzen aus der Folge D von Transaktionen, wobei die großen Sequenzen der Menge Lk jeweils genau k Elemente des Satzes E in einer jeweiligen Reihenfolge R_{Lk} umfassen und ein zugehöriger Stützwert S_{Lk} auf der Folge D von Transaktionen jeweils größer oder gleich dem vorgegebenen Stützwert S ist, wobei beim Ermitteln der Menge Lk nur Sequenzen berücksichtigt werden, die als sich teilweise überlappende Teilsequenzen zwei der großen Sequenzen der Menge Lk-1 mit der jeweiligen Reihenfolge R_{Lk-1} umfassen; und
- Mitteln zum Wiederholen des Schrittes c) für k=k+1 und Abbrechen des Wiederholens des Schrittes c) beim Erfüllen einer vorgegebenen Abbruchbedingung.

Zusammenfassung

Die Erfindung betrifft eine Verfahren und eine Vorrichtung zum Ermitteln einer Menge großer Sequenzen aus einer elektronischen Datenbank mit einer Folge D={d1, ..., dn} von Transaktionen di ($1 \leq i \leq n$) in einem Computersystem mit einem implementierten Abfragemodul, wobei jede der großen Sequenzen auf der Folge D von Transaktionen di einen Stützwert aufweist, der größer oder gleich einem vorgegebenen Stützwert S ist, wobei jede der Transaktionen di der Folge D eine Sequenz von Elementen eines Satzes E={e1, ..., em} von Elementen ej ($1 \leq j \leq m$) ist. Eine Menge Lk ($k > 2$) großer Sequenzen wird aus der Folge D von Transaktionen ermittelt, wobei die großen Sequenzen der Menge Lk jeweils genau k Elemente des Satzes E in einer jeweiligen Reihenfolge R_{Lk} umfassen und ein zugehöriger Stützwert S_{Lk} auf der Folge D von Transaktionen jeweils größer oder gleich dem vorgegebenen Stützwert S ist, wobei beim Ermitteln der Menge Lk nur Sequenzen berücksichtigt werden, die als sich teilweise überlappende Teilsequenzen zwei der großen Sequenzen der Menge Lk-1 mit der jeweiligen Reihenfolge R_{Lk-1} umfassen.